Treball de Fi de Grau

Grau en Enginyeria en Tecnologies Industrials

CAN Music Festival: Orquestra Basada en Microcontroladors PIC18 i un bus CAN

MEMÒRIA

Autor: Joan Calvet i Molinas
Director: Manuel Moreno Eguilaz

Convocatòria: Gener 2016

Escola Tècnica Superior

d'Enginyeria Industrial de Barcelona

Resum

En aquest document es troba el desenvolupament d'una orquestra basada en microcontroladors PIC18 i un bus CAN.

L'objectiu del treball és la creació d'una orquestra musical basada en quatre instruments i un director. Cada instrument està basat en un microcontrolador PIC18F4580 de Microchip amb connexió a un bus CAN. El director d'orquestra és un ordinador personal tipus PC amb un adaptador CAN/USB de Kvaser. El director d'orquestra envia ordres i/o la partitura als diferents instruments per tocar una melodia, que pot estar emmagatzemada en la memòria ROM de cada microcontrolador o en el propi ordinador personal. Es fa servir un bus CAN com a mitjà de comunicació entre els instruments i el director d'orquestra. A més a més, els missatges de bus CAN permeten sincronitzar els instruments per a una correcta interpretació de la partitura per cadascun dels instruments.

Sumari

1. Prefaci

1.1. Origen del projecte

Aquest projecte està basat en dos projectes realitzats prèviament. En primer lloc, el projecte realitzat per un enginyer rus, en Victor Timofeev, que consisteix en sintetitzar amb un microcontrolador PIC16 una melodia mitjançant un sistema operatiu a temps real creat per ell mateix **[1]**. En segon lloc, part del projecte realitzat per en Pere Domenech en el seu PFC **[2]**. Aquest consisteix en l'adaptació del projecte anterior a un microcontrolador PIC18.

1.2. Motivació

La motivació per a la realització d'aquest projecte ha estat el fet de poder aplicar els coneixements d'electrònica i programació en el món de la música, aprenent a utilitzar un sistema operatiu en temps real i a programar en llenguatge C.

A més, part de la feina realitzada en aquest TFG servirà de base en un nou projecte de la assignatura “Projectes I” de l’ETSEIB, en el que els professors Josep Vilaplana i Manuel Moreno pretenen muntar una orquestra fent servir microcomputadors basats en la Raspberry PI 2.

1.3. Requeriments previs

Per a realitzar el projecte, són necessaris els coneixements bàsics d'electrònica digital i de transferència de dades amb busos CAN, així com coneixements de programació amb llenguatge *python.* A més, per a la programació dels microcontroladors és de gran ajuda haver treballat prèviament amb el programari MPLAB i tenir coneixements del llenguatge C.

2. Introducció

2.1. Objectius del projecte

L'objectiu del projecte és la creació d'una orquestra musical. Aquesta orquestra consta de dues parts: els quatre instruments -baix, violí, i dues guitarres- representats per quatre microcontroladors; i el director representat per un programa informàtic que s'executa en un ordinador personal. Així doncs, es vol crear un programa capaç de fer la funció de director d'orquestra comunicant-se a partir d'un bus CAN amb els microcontroladors PIC18.

Per aconseguir l'objectiu, aquest projecte es basa en dos projectes existents prèviament comentats en el prefaci. Així doncs, es parteix d'un sistema operatiu en temps real i del codi d'un programa que permet programar un PIC18 per a reproduir una partitura.

2.2. Abast del projecte

Amb aquest TFG l'orquestra resultant ha de poder:

- Reproduir una melodia curta guardada en memòria ROM de cada microcontrolador de cada instrument.

- Reproduir una melodia curta enviada per bus CAN pel director d'orquestra i emmagatzemada temporalment en la memòria RAM de cada microcontrolador de cada instrument.

- Reproduir una melodia llarga enviada per bus CAN pel director d'orquestra i emmagatzemada parcialment en la memòria RAM de cada microcontrolador de cada instrument amb cues de dades que es van omplint i buidant a mesura que es va reproduint la melodia.

- Generar una partitura compatible amb els instruments a partir d'un fitxer en format midi amb quatre veus.

3. Descripció dels antecedents

Per aquest projecte s'ha partit de dos projectes realitzats anteriorment. El projecte *quartet4* per PIC16 creat per en Victor Timofeev i el projecte de fi de carrera d'en Pere Domenech - sent el segon una adaptació del primer.

Així doncs, per poder entendre el desenvolupament del software d'aquest projecte, és imprescindible entendre els fonaments dels dos projectes esmentats.

3.1. Projecte "Quartet4" per PIC16

Aquest projecte és un exemple d'aplicació del sistema operatiu en temps real creat pel mateix autor, Victor Timofeev. Consisteix en sintetitzar una melodia a partir d'unes notes emmagatzemades en la memòria ROM d'un microcontrolador.

El sistema operatiu *OSA* és un sistema operatiu en temps real de multi tasca dissenyat per a diferents tipus de microcontroladors PIC. El sistema operatiu s'encarrega de gestionar els recursos del sistema i executar les diferents tasques definides en el programa en funció de les prioritats de cada una.

Una tasca en *OSA-RTOS* és una funció en C que ha de contenir un bucle infinit. Dins d'aquest bucle es codifica la tasca concreta que es vulgui realitzar.

El programa creat, genera quatre canals d'àudio utilitzant una sortida PWM amb resolució de 8 bits a 78 kHz. L'esquema del sintetitzador és el mostrat a la Figura 1.

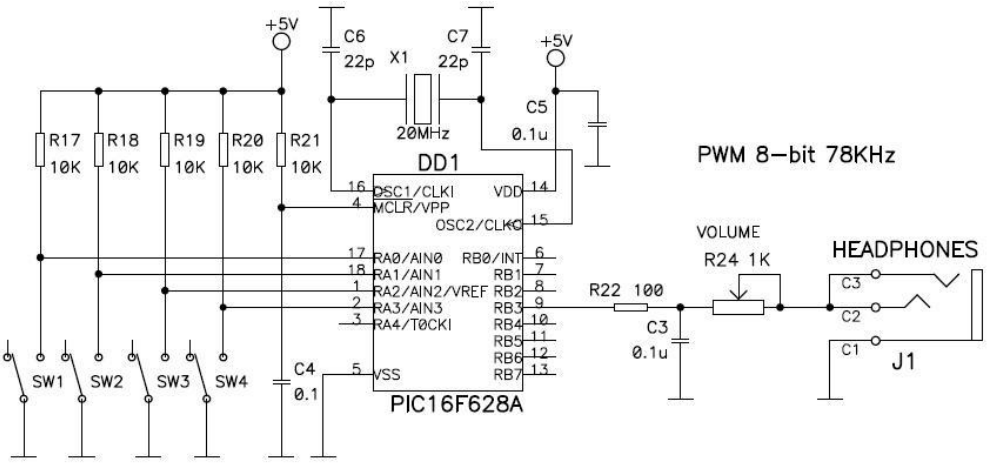

Figura 1: Esquema del circuit físic emprat per al "quartet4". Font: http://www.pic24.ru/doku.php/en/osa/ref/appendix/quartet

La part del codi està composta per diferents arxius que permeten aconseguir l'objectiu de sintetitzar la melodia. A continuació s'explica un breu resum de la funció de cada arxiu.

- *quartet_main.c*: És el fitxer que conté el programa principal. En aquest fitxer es criden la resta d'arxius i és el que executa el software. Serà utilitzat per aquest projecte a partir de l'adaptació explicada en el capítol 3.2

- *interrupt.c*: Aquest arxiu conté el codi corresponent al sintetitzador.

- *freqs.c*: Conté les taules de freqüències corresponents a un rang de notes definit per l'autor. Concretament des d'un *do* a 65.40639 Hz (octava 0) fins a un *sol* sostingut quatre octaves més agut a 1661.21879 Hz (octava 4). A més s'hi especifica el significat de les funcions que es troben en els fitxers partitura.

- *sinus.c*: Conté els valors de les funcions envoltants dels sinus corresponents al so dels tres instruments emprats: violí, baix i guitarra.

- *bach1067.c*: Conté la llista de comandes dels quatre instruments de la partitura referent al *Bourreé* I de la *Suite* No.2 en *si* menor, BWV 1067 - de J.S. Bach. Hi ha una llista de comandes per a cada instrument; baix, violí, guitarra 1 i guitarra 2. Les comandes emprades per a definir una partitura són creades pel mateix autor. Per a la creació de partitures en aquest projecte es partirà d'aquest sistema de comandes.
  - *play(nota, durada):* Indica la nota a reproduir en la octava corresponent i la durada d'aquesta. La nota s'introdueix amb el sistema de notació musical anglès; i els sostinguts s'introdueixen amb el símbol '_' seguint la nota. La durada és un valor enter entre 1 i 4. Exemple: *play(b1_,3).*
  - *pause(durada):* Indica que hi ha un silenci i la seva durada.
  - *setbase(octava):* Indica a partir de quina octava es parteix. Així, en el violí podria aparèixer *setbase(2)* i en el baix *setbase(0).*
  - *repeat(vegades):* Indica que hi ha una repetició de totes les comandes anteriors fins a arribar a una comanda *repeatmarker()* un nombre de vegades concret.
  - *repeatmarker():* Indica que hi ha una marca de repetició. Si posteriorment hi ha una comanda *repeat()*, tornarà en aquest punt.
  - *stop():* Indica el final de la partitura. En el cas del projecte original *quartet4* la comanda *stop()* és sinònim de repetició; ja que el programa torna a reproduir la melodia un cop s'hi arriba.

- *elochka.c*: Conté la definició de les funcions implementades per a l'execució de la partitura.
    - *play(nota,durada):* Genera un nombre binari a partir de l'operació OR lògica entre un nombre creat a partir de l'equivalent a restar 1 a la durada desplaçat 5 bits a l'esquerra i un nombre corresponent a la nota. Aquest nombre s'adjudica en funció de la posició de la nota. El valor 0 correspon a la nota més baixa c0, i s'augmenta en ordre ascendent a l'escala. Així, havent-hi dotze notes per octava considerant també els sostinguts, el valor que correspon a *c1* és 12.
    - *pause(durada):* Genera un nombre a partir de l'operació OR lògica entre el nombre 0x80 i la durada menys 1.
    - *repeat(vegades):* Genera un nombre a partir de l'operació OR lògica entre el nombre 0xC0 i el nombre de vegades.
    - *repeatmarker():* Genera el nombre 0xC0.
    - *setbase(octava):* Genera un nombre a partir de l'operació OR lògica entre el nombre 0xA0 i el nombre equivalent a multiplicar 12 per el valor de l'octava.
    - *stop():* Genera el nombre 0xE0.

```
repeatmarker(),   0xC0    -64    11000000
setbase(0),       0xA0    -96    10100000
pause(4),         0x83   -125    10000011
play(b0, 1),      0x0B     11    00001011
play(c1_,1),      0x0D     13    00001101
play(d1, 2),      0x2E     46    00101110
play(c1_,2),      0x2D     45    00101101
```

Figura 2: Correspondència entre les comandes de la partitura del baix i la seva traducció a bits. Font pròpia.

- *osa.c*: Conté les variables del sistema operatiu i la definició de les funcions.

3.2. Projecte de Fi de Carrera d'en Pere Domenech

La part que s'ha considerat del projecte de Pere Domenech és la migració del codi del *quartet4* a un PIC18.

Per poder entendre el projecte que s'explica posteriorment, és necessari entendre el funcionament del codi; d'aquesta manera, a continuació s'explica un resum a partir del qual s'ha iniciat aquest treball.

El fitxer que s'explica amb detall és el del programa principal, ja que és el que s'ha hagut de modificar. Aquest és el fitxer *quartet_main.c*.

En primer lloc, el programa inicialitza totes les variables existents en altres documents i necessàries en aquest.

El programa consta de cinc tasques; quatre referents als diferents instruments i una tasca anomenada *conductor* que fa el paper de director. Cada tasca de cada instrument s'encarrega d'iniciar el so corresponent a l'instrument indicat a partir de la veu de la partitura adequada emmagatzemada en memòria ROM. A partir de la funció *NoteWork* els instruments interpreten el significat de cada comanda rebuda de la partitura i avancen en la partitura. La funció del director, en aquest cas la tasca *conductor*, és la de sincronitzar els instruments. Un cop s'ha reproduït la partitura, el director s'encarrega d'iniciar alhora els quatre instruments.

La funció *main*, que serà la funció amb més modificacions en el desenvolupament d'aquest TFG, s'encarrega d'iniciar el sistema operatiu, de crear les tasques i finalment d'executar-lo.

Per a crear els senyals de les veus dels instruments, el programa realitza un seguit d'operacions amb els valors dels sinus continguts en l'arxiu *sinus.c*; diferents per a cada instrument.

En definitiva, el programa funciona de la següent manera: La funció *main* inicia el sistema operatiu i crea les cinc tasques seguint un ordre de prioritats. Primer els quatre instruments i després la tasca *conductor*. Un cop creades s'executa el sistema operatiu. Les tasques realitzen indefinidament la seva funció mitjançant un bucle infinit. En cas d'estar aturat el so de l'instrument, les tasques de cada un l'inicien mitjançant la funció *InitSoundVariable* a partir del so i la llista de notes corresponent. El so s'obté a partir d'un seguit d'operacions matemàtiques concretes amb els valors de les funcions sinus descrits en el fitxer *sinus.c* i el llistat de notes s'obté a partir de la traducció de la partitura en el fitxer *bach1067.c* amb el fitxer *elochka.c*. A partir d'aquí, la funció encarregada d'interpretar les dades del llistat de notes és la funció *NoteWork*. Aquesta funció tradueix les comandes rebudes i executa l'ordre corresponent.

Així doncs, es parteix d'un programa que realitza la feina del director i dels instruments a la vegada. La feina de director i dels quatre instruments la realitza el mateix microcontrolador, per tant el programa s'ha de dissenyar per programar els quatre instruments. Llavors, per a la realització d'aquest TFG és necessari realitzar canvis en el programa principal *quartet_main.c* per aconseguir programar un sol instrument en el microcontrolador, i rebre les ordres d'un director extern.

4. Proposta de hardware d'àudio

El director de l'orquestra es basa en un programa executable des d'un ordinador personal. Així doncs, el hardware necessari per al projecte és el referent als instruments. Els microcontroladors utilitzats per fer la feina d'instruments estan basats en plaques ja existents. Les plaques que es fan servir són propietat de la secció Sud del Departament d'Enginyeria Electrònica de la UPC. Estan basades en un microcontrolador PIC18F4580 de 8 bits amb controlador de bus CAN integrat i es feien servir fa uns quants anys en diferents assignatures del pla antic.

Amb aquestes plaques no és possible efectuar la reproducció del so ja que no disposen d'altaveus. Per tant, s'ha de crear un hardware d'àudio i integrar-lo a la placa, no només per a poder reproduir el resultat final, sinó també per a poder realitzar comprovacions del funcionament durant el disseny del programa. Aquest hardware és un simple filtre passa baixos de primer ordre que permet convertir la sortida PWM del microcontrolador en un senyal analògic que conté el so en qüestió que s'està reproduint.

El hardware a realitzar es basa en el que es veu en la part inferior dreta de la Figura 1.

La principal diferència entre els dos hardwares és que en el creat no s'ha utilitzat un potenciòmetre com a regulador del volum del so, sinó que es fixa el valor d'una resistència havent fet proves amb diferents valors (Figura 3).

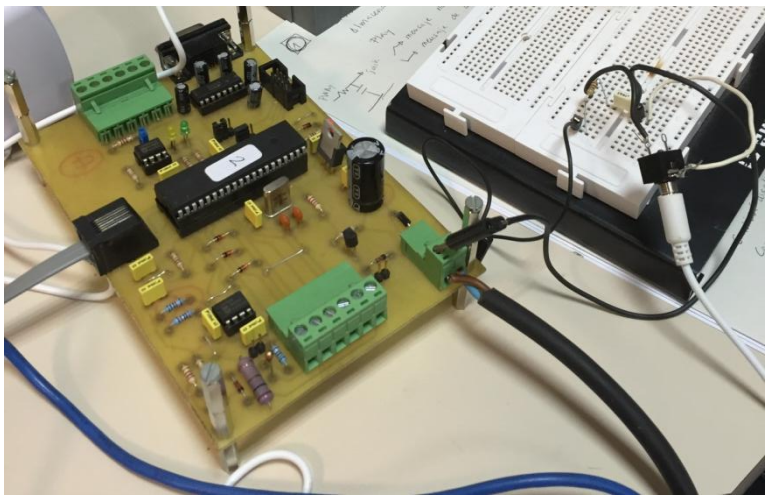

Figura 3: Circuit de prova per variar el valor de la resistència. Font pròpia.

Finalment, s'ha establert un circuit amb dues resistències de 100Ω i 22kΩ, un condensador de 100nF i un jack aeri per a connectar els altaveus; tal i com es mostra en la Figura 4.

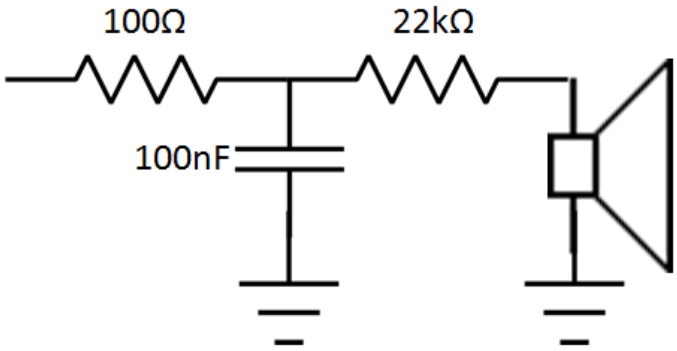

Figura 4: Circuit a implementar per al hardware d'àudio. Font pròpia.

4.1. Fabricació del hardware

Per crear la placa de hardware amb el circuit necessari en primer lloc s'han obtingut quatre retalls d'una placa de les mides suficients per a poder-los incorporar a la placa del microcontrolador (Figura 5).

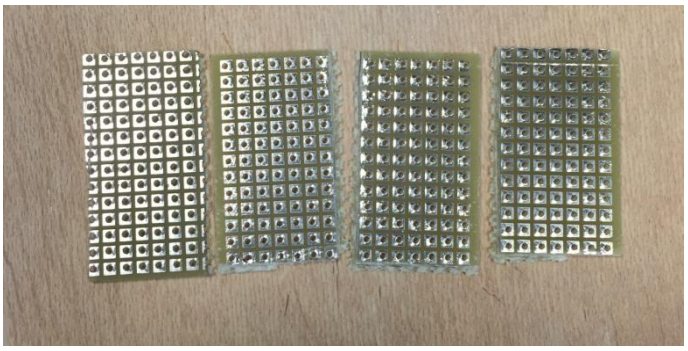

Figura 5: Retalls de placa base per al hardware. Font pròpia.

A continuació s'ha foradat la placa per a poder afegir-hi el jack aeri, i seguidament s'han introduït els components amb els valors específics descrits anteriorment (Figura 6).

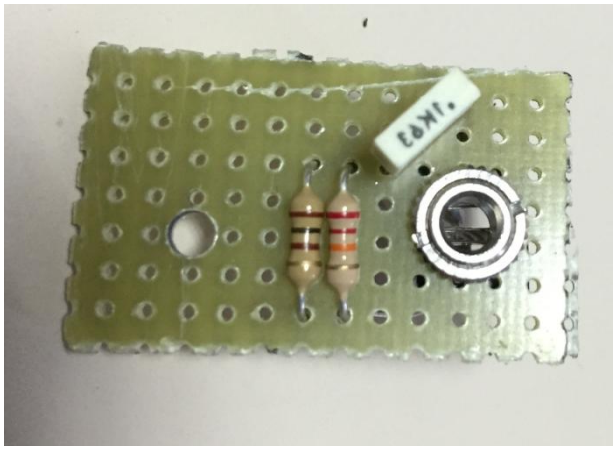

Figura 6: Placa de hardware amb components i forat per a collar. Font pròpia.

Un cop afegits els components, s'han soldat les connexions descrites a la Figura 4 amb els punts corresponents a la placa del microcontrolador i després s'ha collat amb un cargol a la respectiva placa. A la Figura 7 es mostra el resultat d'haver afegit el hardware d'àudio a la placa original.

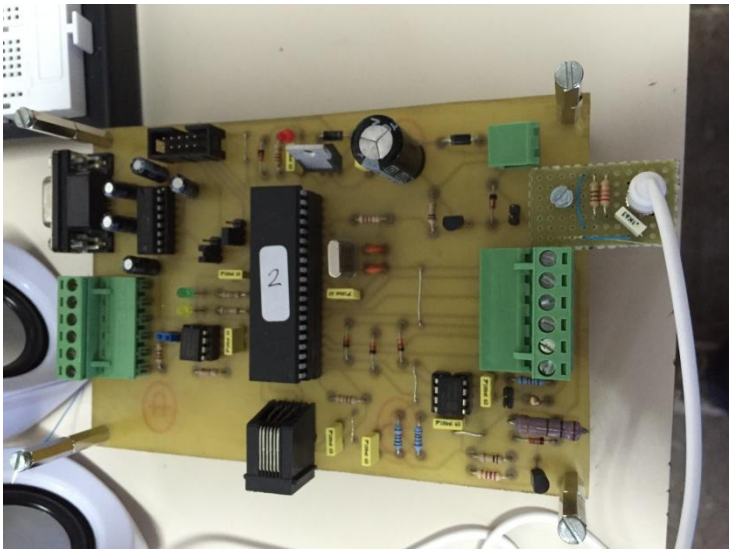

Figura 7: Hardware d'àudio afegit a la placa del microcontrolador. Font pròpia.

Per acabar, s'ha afegit a la part superior de la placa del microcontrolador una planxa de metacrilat per fer de suport dels altaveus i s'han afegit els altaveus a cada una de les plaques (Figura 8).

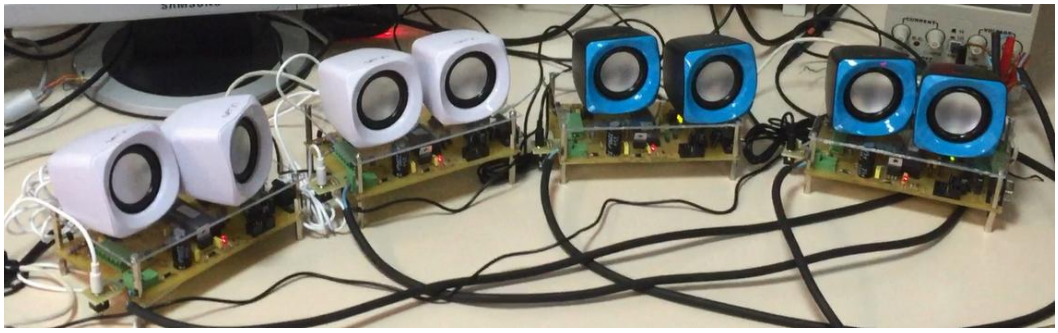

Figura 8: Plaques definitives amb els altaveus afegits. Font pròpia.

Els altaveus utilitzats s'alimenten mitjançant un cable USB que per minimitzar espai i problemes de cables es pot introduir en l'ordinador que s'utilitzi.

Un cop obtingut tot el hardware necessari per a la comprovació i validació del funcionament del software comença la part més densa del TFG: el desenvolupament d'aquest.

5. Desenvolupament del software

El software creat per el projecte presenta dues grans branques: la programació en llenguatge *python* del programa director (PC) i la programació en llenguatge C dels instruments (PIC18F4580) a partir dels antecedents comentats anteriorment.

S'han realitzat tres versions diferents de software. La primera versió, i per tant la més senzilla, consisteix en la creació d'un director que doni l'ordre de començar (*start*) a reproduir una partitura emmagatzemada en memòria ROM per cada instrument. La segona consisteix en dissenyar el director de manera que enviï la partitura corresponent a cada instrument. Aquests emmagatzemen la partitura rebuda en memòria RAM, i un cop s'ha enviat tota la partitura el director dóna l'ordre de començar a reproduir. Evidentment la partitura haurà de ser curta perquè càpiga en la memòria RAM de cada microcontrolador. La versió definitiva, és a dir, la tercera versió del software consisteix en programar el director de manera que un cop enviada una part de la partitura doni l'odre de reproduir i vagi enviant la resta de la partitura a mesura que es vagi necessitant, comunicant-se amb els microcontroladors mitjançant un bus CAN. Aquesta darrera versió permet reproduir partitures de qualsevol longitud.

5.1. Partitura en ROM

Aquesta versió més senzilla ha servit d'instrument per a la comprovació del correcte funcionament del programa a adaptar creat en un projecte anterior i comentat en el capítol 3.2.

El programa director envia amb un missatge de bus CAN l'ordre d'inici de la reproducció. Els instruments reprodueixen una partitura emmagatzemada prèviament en memòria ROM.

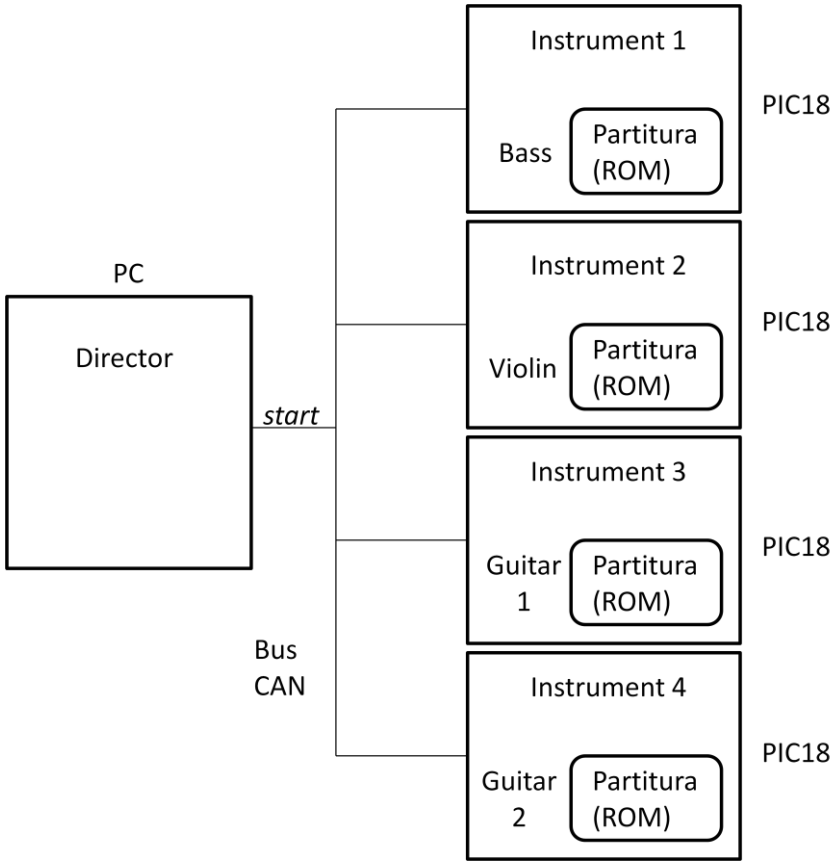

Figura 9: Esquema de funcionament. Font pròpia.

5.1.1. Director

La versió del director és senzilla. Un cop els instruments estan a punt, és a dir, engegats, s'executa el programa director de manera que envia un missatge amb l'ordre de començar (*start*) mitjançant un bus CAN.

Aquest missatge consta d'un identificador global i el missatge que indica l'ordre.

```
def start(self):
        stat = canl2.canSetBusParams(self.handle, self.baud_rate, 0, 0, 0, 0, 0)

        print (stat)

        if stat==canl2.canOK:

                print("CAN parameters OK")

                stat = canl2.canBusOn(self.handle)
                if stat==canl2.canOK:

                        print("Bus ON")

                        try:
                                sleep(0.01)  # sleep 10ms

                                stat = canl2.canWrite(self.handle, self.iden_global, self.start_message, 2, canl2.canMSG_STD)


                        except KeyboardInterrupt:
                                print("Game Over")
                else:
                        print("Error Bus OFF")
        else:
                print("Error Bus Parameters")
```

Figura 10: Codi corresponent a la primera versió del director. Font pròpia.

Es pot observar en la figura superior que un cop comprovats els paràmetres del bus CAN, s'escriu un missatge d'inici (*start message*) amb l'identificador global. Tant l'identificador com el missatge concret s'especifiquen en la inicialització del director.

5.1.2. Quartet

La versió del quartet és molt similar a la versió ja existent d'en Pere Domenech [pàg. 11].

El canvi més important que s'ha mantingut al llarg de les diferents versions és el de dissenyar el programa de manera que només es programi un instrument cada vegada. Per fer-ho s'ha creat una nova tasca anomenada *Task_INS* que realitza la funció que feien abans les tasques dels quatre instruments. Al voler programar un sol instrument a cada microcontrolador, només és necessari la feina amb les notes i el so corresponents a aquell instrument.

Al fitxer de configuració *Config.h* s'hi especifica l'instrument que es vol programar i a l'hora d'executar, s'utilitzaran unes funcions o altres en funció del que s'hagi seleccionat. Això s'ha aconseguit mitjançant la directiva de C *#define* més el nom de l'instrument, que permet definir el nom d'aquest i adaptar les funcions al que s'hagi definit mitjançant la directiva *#ifdefined* més el nom del mateix instrument. Per al correcte funcionament del programa no pot estar més d'un instrument definit a la vegada, de manera que en la configuració només hi pot haver un *#define* definit i la resta han d'estar en forma de comentari. Per evitar errors

d'aquest tipus s'ha afegit un detector que retorni error en cas d'haver definit més d'un instrument (Figura 11).

```
//#define Bass
//#define Violin
//#define Guitar1
#define Guitar2

#if defined(Bass)&& defined(Violin)
    #error "Only one instrument allowed"
#endif

#if defined(Bass)&& defined(Guitar1)
    #error "Only one instrument allowed"
#endif

#if defined(Guitar1)&& defined(Violin)
    #error "Only one instrument allowed"
#endif

#define START_MESSAGE 0x01

#define IDEN_GLOBAL 0x55
#define IDEN_0  0x100
```

Figura 11: Codi corresponent al document de configuració. Cas de definició de la segona guitarra. Font pròpia.

```
void Task_INS (void)
{
    S.bStopped = 1;
    for (;;) {
        if (S.bStopped) {
            // Tell to conductor, that notelist over
            OS_Flag_Set_0(flag_Playing, FLAG_INS_PLAYING);
            // Wait for command to start playing from conductor
            OS_Bsem_Wait(BS_START_MUSIC);
            // Re-init channel data
            #ifdef Bass
                  InitSoundVariable(&S, notelist_bass);
            #endif

            #ifdef Violin
                  InitSoundVariable(&S, notelist_violin);
            #endif

            #ifdef Guitar1
                  InitSoundVariable(&S, notelist_guitar1);
            #endif

            #ifdef Guitar2
                  InitSoundVariable(&S, notelist_guitar2);
            #endif

            OS_Flag_Set_1(flag_Playing, FLAG_INS_PLAYING);
            // Retranslate command to next task
            OS_Bsem_Set(BS_START_MUSIC);
        }

        // Wait for command from conductor
        do {
            OS_Bsem_Wait(BS_INS);
        } while (S.cDuration--);

        // Read next note and set new duration
        NoteWork(&S);
    }
}
```

Figura 12: Codi de definició de la tasca instrument. Font pròpia.

Per tal d'esperar el missatge de bus CAN que permet iniciar la reproducció de la partitura en ROM, s'ha creat un bucle al programa *main* de manera que no comença a reproduir fins que es rebi el missatge específic des del director, tal i com es mostra a la Figura 13.

```
CANInitialize(1, 5, 7, 6, 2, CAN_CONFIG_VALID_STD_MSG);
while (1) {

if(CANIsRxReady())
{
    if(CANReceiveMessage(&identificador, MensajeRecibido, &LongMensaje, &FlagsMensaje))
    {
        if(identificador == IDEN_GLOBAL)
        {
            if (MensajeRecibido[0] == START_MESSAGE)
            {
                tempo = MensajeRecibido[1];
                break;
            }
        }

    }
}
```

Figura 13: Bucle infinit contingut en la funció *main* per a rebre missatge d'inicialització. Font pròpia.

Com es pot observar en la figura superior, en el missatge d'inici s'hi envia el tempo a seguir en la reproducció de la peça. Així doncs, el tempo el determina el programa director creat i no es defineix en la programació dels instruments.

Llavors, un cop es rep el missatge específic, el sistema operatiu entra en funcionament i reprodueix la partitura emmagatzemada.

5.2. Partitura completa en RAM

Aquesta versió és útil per a reproduir partitures curtes. En el director s'hi especifiquen les quatre veus de la partitura a reproduir, i aquest les envia mitjançant el bus CAN cap als instruments. Aquests, guarden la partitura corresponent en memòria RAM i comencen a reproduir un cop el director doni l'ordre d'inici.

Per tant, si la partitura a enviar és massa llarga, podria ser que s'hagués d'esperar massa fins a haver-la guardat tota, o fins i tot que els instruments no la poguessin emmagatzemar sencera. Així, la utilitat d'aquesta versió està limitada a partitures curtes.

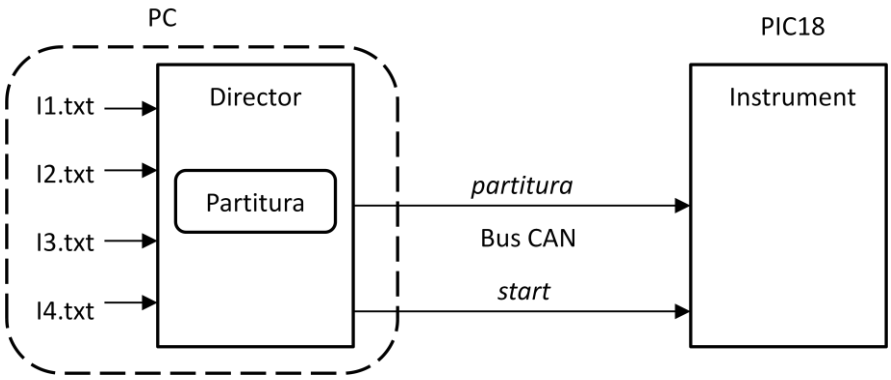

Figura 14: Esquema de funcionament. Font pròpia.

5.2.1. Director

A partir de quatre fitxers de tipus *.txt* el director envia a partir d'un bus CAN la partitura als quatre instruments. Un cop s'ha enviat la partitura sencera, el director envia un missatge d'inicialització que permet als instruments començar la reproducció.

Els fitxers de text que s'introdueixen al director tenen un format específic. Per simplificar la feina, s'ha optat per mantenir l'estructura dissenyada per l'enginyer rus Victor Timofeev **[1]** en les seves partitures. Aquests fitxers indiquen les ordres que han de rebre els instruments per a la correcta reproducció de la partitura.

Així doncs, el programa director ha de ser capaç de llegir aquests fitxers i traduir-los a bits de manera que siguin interpretables pels instruments. Aquesta traducció es basa en la que es realitza en el fitxer de *elochka.c* d'en Victor Timofeev.

Un cop traduïda la partitura, el director ha d'enviar les comandes corresponents a cada instrument. Per fer-ho, s'envia un missatge amb un identificador comú per a tots els instruments i quatre ordres; una per a cada un. Aquest missatge s'obté a partir de les llistes de comandes obtingudes a partir de les partitures llegides.

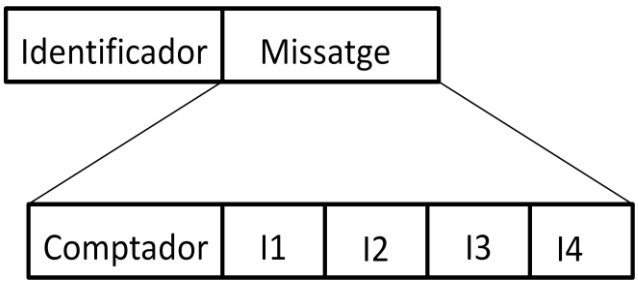

Figura 15: Esquema del missatge enviat. Cada instrument n'agafa l'ordre corresponent I que correspongui. Font pròpia.

```
def create_frame(self):
        self.frame[0] = self.counter
        for i in range(1,5):
                try:
                        self.frame[i] = eval('self.rom' + str(i) + '.pop(0)')
                except:
                        self.frame[i] = 0
```

Figura 16: Codi de la funció que genera el missatge a enviar als quatre instruments. Font pròpia.

Com que existeix la possibilitat que les partitures de diferents instruments tinguin llargàries diferents, un cop s'ha llegit la partitura d'un instrument sencera, el missatge enviat per a aquell instrument és un 0.

Un cop s'han enviat les ordres de tots els instruments, el director envia un missatge d'inicialització que permet l'inici de la reproducció.

5.2.2. Quartet

La segona versió del quartet és una adaptació de l'anterior. Es conserva el fet de programar un sol instrument i no els quatre a la vegada.

Al no tenir una partitura definida ja emmagatzemada, el programa no disposa de dades a processar fins que el director les enviï. Per obtenir la partitura enviada pel director, el programa *main* es modifica. Es crea un bucle infinit que llegeix missatges rebuts amb el bus CAN de manera que en funció de l'instrument que s'ha decidit programar agafa una part del missatge o un altre. Part d'aquest missatge es quadra en una cua a la memòria RAM de l'instrument per a poder ser processat posteriorment.

```
CANInitialize(1, 5, 7, 6, 2, CAN_CONFIG_VALID_STD_MSG);
while (1) {
if(CANIsRxReady())
{
    if(CANReceiveMessage(&identificador, MensajeRecibido, &LongMensaje, &FlagsMensaje))
    {
        if(identificador == IDEN_GLOBAL)
        {
            if (MensajeRecibido[0] == START_MESSAGE)
            {
                tempo = MensajeRecibido[1];
                break;
            }
        }
        if(identificador == IDEN_0)
        {
            #ifdef Bass
                  cola_rx[i++] = MensajeRecibido[1];
            #endif

            #ifdef Violin
                  cola_rx[i++] = MensajeRecibido[2];
            #endif

            #ifdef Guitar1
                  cola_rx[i++] = MensajeRecibido[3];
            #endif

            #ifdef Guitar2
                  cola_rx[i++] = MensajeRecibido[4];
            #endif
        }
    }
}
```

Figura 17: Codi de la funció *main*. Emmagatzematge del missatge rebut. Font pròpia.

Així doncs, la feina realitzada en la funció *NoteWork* no es basa en les comandes emmagatzemades en memòria ROM, sinó que la tasca de l'instrument crida la funció a partir de les dades emmagatzemades prèviament en RAM.

En el moment en què es rep el missatge d'iniciar, és a dir, que ja s'han enviat les partitures a tots els instruments, se surt del bucle i comença la reproducció dels instruments.

5.3. Partitura en temps real

Aquesta versió permet reproduir partitures de qualsevol grandària. Partint de la versió anterior, un cop traduïda la partitura, el director envia un determinat nombre de missatges a tots els instruments (només una part de la partitura) a partir de la funció *start* definida en la versió anterior.

Un cop enviats aquests missatges, s'envia l'ordre d'inicialització (*Play*), de manera que els instruments comencen a reproduir els missatges ja rebuts. A partir d'aquí, el director va consultant amb els instruments si necessiten més comandes o no, i va enviant els missatges a cada instrument en particular.

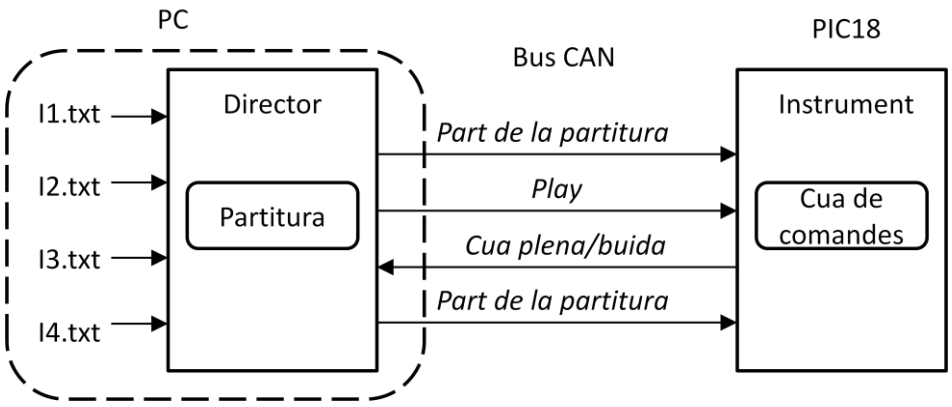

Figura 18: Esquema de funcionament. Font pròpia.

5.3.1. Director

El director conserva la base de la versió anterior. A partir dels quatre fitxers de text que es vulguin reproduir, es generen les ordres que s'hauran d'enviar als diferents instruments.

En les versions anteriors s'executava el programa director després d'haver engegat els microcontroladors. En aquesta versió el procediment és diferent: primer de tot s'executa el director. Un cop ha traduït les partitures com en la versió anterior, el director espera a rebre un missatge ("Alive") de cada instrument per a poder començar a enviar les partitures. D'aquesta manera s'evita perdre algun missatge a causa d'algun error en els microcontroladors o en el bus.

Un cop ha rebut un missatge de cada instrument confirmant el seu bon funcionament, s'envia un seguit de missatges per als quatre instruments amb un identificador en comú (identificador global). El nombre de missatges a enviar és una variable que es pot modificar segons la grandària de la partitura.

Un cop enviats aquests missatges, el director envia un missatge d'inicialització. D'aquesta manera els instruments començaran a reproduir els missatges rebuts fins al moment.

A partir d'aquí el programa director i els instruments interactuen en temps real. Fins que no s'han enviat les quatre partitures senceres, el director i els instruments es comuniquen enviant-se missatges.

Com que cada instrument utilitza els missatges rebuts a un ritme diferent en funció de com és la partitura per a cadascun, el director ha de consultar l'estat d'aquests per separat; és a dir, cada missatge enviat té un identificador diferent en funció de a qui vagi dirigit.

Els instruments omplen i buiden una cua a mesura que reben missatges i els reprodueixen. El director comprova si la cua d'algun instrument està plena per decidir si s'han d'enviar més

missatges a aquest instrument. Si rep un missatge de cua plena d'un instrument el director deixa d'enviar-li missatges fins a rebre un missatge de cua buida d'aquest.

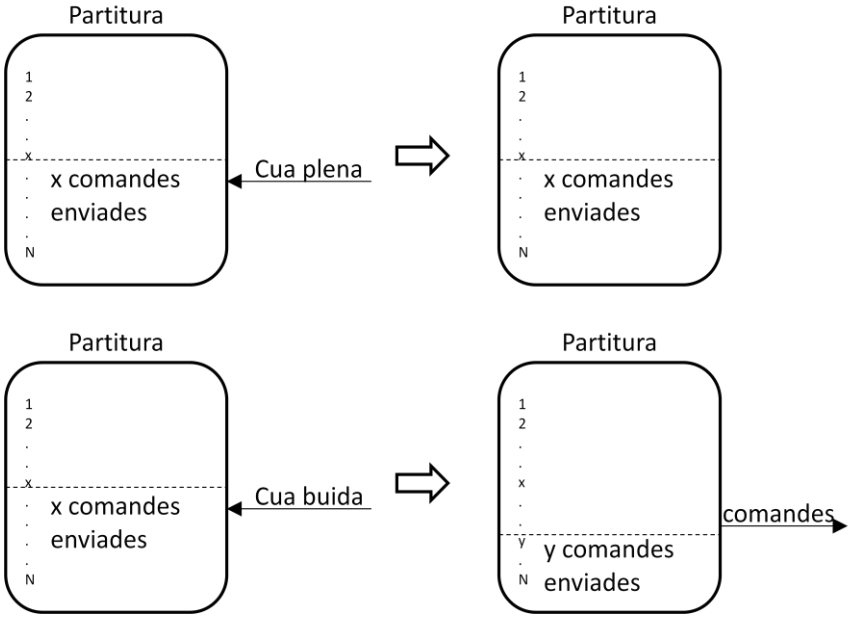

Figura 19: Esquema de funcionament de l'enviament de comandes de la partitura. Font pròpia.

5.3.2. Quartet

Per l'última versió del quartet s'han hagut d'utilitzar els serveis del sistema operatiu OSA RTOS i crear una nova tasca en el programa.

En primer lloc es manté el fet de dissenyar el programa per a un sol instrument. El següent punt a tenir en compte és la modificació del *main*. Com s'ha explicat en el punt anterior, el director espera a saber que els microcontroladors estan en funcionament per a poder enviar la partitura. D'aquesta manera, d'entrada s'envia un missatge repetit tres vegades amb el bus CAN cap al director amb un missatge que demostri que l'instrument està actiu (*Alive*). Un cop enviat el missatge, s'entra en un bucle en què fins que no es rebi l'ordre de començar a reproduir s'afegeixen les comandes rebudes en una cua de missatges.

```
    Message[0] = ALIVE_MESSAGE;
#ifdef Bass
    iden_s = IDEN_BASS;
#endif
#ifdef Violin
    iden_s = IDEN_VIOLIN;
#endif
#ifdef Guitar1
    iden_s = IDEN_GUITAR1;
#endif
#ifdef Guitar2
    iden_s = IDEN_GUITAR2;
#endif

    while(!CANIsTxReady());
    CANSendMessage(iden_s, Message, 2, CAN_TX_PRIORITY_0 & CAN_TX_STD_FRAME & CAN_TX_NO_RTR_FRAME);
    while(!CANIsTxReady());
    CANSendMessage(iden_s, Message, 2, CAN_TX_PRIORITY_0 & CAN_TX_STD_FRAME & CAN_TX_NO_RTR_FRAME);
    while(!CANIsTxReady());
    CANSendMessage(iden_s, Message, 2, CAN_TX_PRIORITY_0 & CAN_TX_STD_FRAME & CAN_TX_NO_RTR_FRAME);
// enviar missatge alive
```

Figura 20: Codi de la funció *main* on s'envia el missatge d'instrument actiu. Font pròpia.

Un cop se surt del bucle, el programa comença a reproduir els missatges rebuts com s'havia fet en les versions anteriors.

Per poder enviar els missatges de cua plena o cua buida al director, s'ha creat una nova tasca en el programa anomenada *Task_comCAN*. Aquesta tasca va afegint els nous missatges rebuts a la cua a més de dedicar-se a comprovar l'estat de la cua de comandes. Si la cua està plena, és a dir, el valor de les comandes a interpretar arriba al valor de cua màxima especificat pel director, s'envia un missatge de cua plena al director. Arribats a aquest punt, la tasca de comunicació per CAN, espera fins a haver buidat la cua fins a un mínim per a tornar a demanar missatges. Això s'ha aconseguit amb una funció pròpia del sistema operatiu anomenada *OS_Cond_Wait.* Quan es compleix la condició -en aquest cas que hi hagi menys d'un valor de comandes a la cua- l'instrument envia el missatge d'avís de cua buida cap al director.

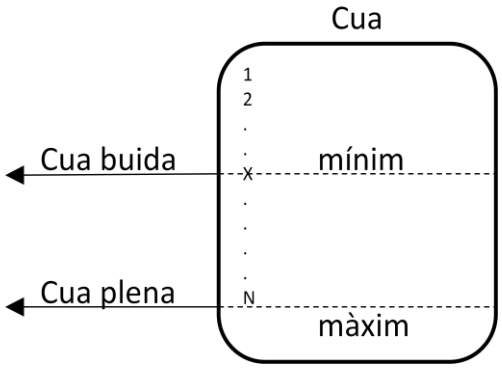

Figura 21: Esquema de funcionament de la cua de comandes en l'instrument. Font pròpia.

Per facilitar la feina a l'hora de fixar constants, s'ha creat un fitxer de configuració -anomenat *Config.*h- on s'hi defineix el tipus d'instrument a programar, així com el valor mínim de la cua i els valors dels missatges de cua buida, de cua plena, i d'instrument en funcionament. També s'ha creat un fitxer de definició dels identificadors a utilitzar en cada missatge, anomenat *iden.h*.

```
void Task_comCAN (void)
{
#ifdef Bass
    iden_s = IDEN_BASS;
    iden_rx = IDEN_RX_B;
#endif
#ifdef Violin
    iden_s = IDEN_VIOLIN;
    iden_rx = IDEN_RX_V;
#endif
#ifdef Guitar1
    iden_s = IDEN_GUITAR1;
    iden_rx = IDEN_RX_G1;
#endif
#ifdef Guitar2
    iden_s = IDEN_GUITAR2;
    iden_rx = IDEN_RX_G2;
#endif
    for (;;){
    OS_Cond_Wait((RXB0CONbits.RXFUL | RXB1CONbits.RXFUL))  // Wait for CAN message
    if(CANReceiveMessage(&iden, message_rx, &LongMensaje,   &FlagsMensaje))
    {
        if (iden == iden_rx)
        {
                counter = counter + 1;
                cola_rx[i++] = message_rx[0];
                if (counter >= QUEUE_MAX)
                {
                    Message[0] = FULL_MESSAGE;
                    while(!CANIsTxReady()); // Mentre no hi hagi cap buffer lliure -> esperem
                    CANSendMessage(iden_s, Message, 2, CAN_TX_PRIORITY_0 & CAN_TX_STD_FRAME & CAN_TX_NO_RTR_FRAME);
                    OS_Cond_Wait(counter<=QUEUE_MIN);
                    Message[0] = EMPTY_MESSAGE;
                    while(!CANIsTxReady());
                    CANSendMessage(iden_s, Message, 2, CAN_TX_PRIORITY_0 & CAN_TX_STD_FRAME & CAN_TX_NO_RTR_FRAME);
                }
        }
    }
}
}
```

Figura 22: Codi de la tasca de comunicació dels instruments amb el director via el bus CAN. Font pròpia.

5.4. Generació de partitures

Per a poder reproduir més melodies a part de la ja definida en la creació del projecte *quartet4* de Victor Timofeev, s'ha decidit escriure un programa que generi partitures en el format necessari per a l'execució del programari realitzat prèviament.

Per aconseguir l'objectiu s'ha partit d'un software desenvolupat per el reconegut MIT **[3]**, anomenat *music21* que permet obtenir un extens seguit de dades a partir d'un fitxer de tipus midi.

S'ha partit d'un programa en *python* desenvolupat pel professor Manuel Moreno Eguilaz anomenat *Partiture.py* que genera fitxers de text amb informació de les partitures a partir d'un fitxer tipus midi utilitzant les llibreries obtingudes de *music21*. El codi del programa realitzat pel professor Manuel Moreno i el del programa creat per aquest TFG es troben adjunts en l'annex B.

En aquests fitxers s'hi troba l'instant de reproducció d'una nota, el valor numèric de la nota i la seva duració.

Per tant, s'ha creat un programa que a partir dels fitxers creats amb el programa *Partiture.py* generi fitxers en el format necessari per poder ser utilitzats amb el programa director creat prèviament.

Aquest programa creat adequa les partitures al format del qual s'ha partit a partir dels antecedents ja comentats. Per tant, a partir de realitzar canvis en el programa de programació dels microcontroladors es podria arribar a realitzar una millora substancial en la capacitat de la generació de les partitures i la capacitat de l'orquestra per reproduir-les. A l'hora de crear noves partitures s'ha de comprovar en primer lloc que aquestes siguin fàcilment adaptables. Per exemple, el programa no admet la reproducció de més d'una nota a la vegada amb la mateixa veu, per tant, s'ha de tenir en compte que les peces en què les alguna veu hagi de reproduir un acord presentaran problemes de reproducció.

Amb tot això, s'encoratja a futurs projectes relacionats a treballar sobre el programari creat per tal de millorar-lo.

6. Test i validació

Per comprovar el correcte funcionament del software i el hardware utilitzats s'ha provat amb dues partitures diferents.

La primera ha estat la partitura ja creada per en Victor Timofeev del *Bourreé I* de la *Suite* No 2 en *si* menor de Johann Sebastian Bach. Per poder comprovar també el funcionament de la generació de noves partitures a partir de fitxers midi, s'ha obtingut el fitxer midi de la pàgina web *https://musescore.org/ca* de la cançó *Air in the G String* per a quatre veus; un arranjament del segon moviment de la *Suite* No 3 en *re* major, BWV 1068 de Bach fet per el violinista alemany August Wilhelmj.

S'ha observat la correcta reproducció, és a dir, el sincronisme i la reproducció adequada de cada una de les veus de les dues peces; i s'ha donat per validat el funcionament del projecte.

Per comprovar la sincronització dels instruments durant la realització del projecte s'han anat realitzant diferents proves amb les peces. S'ha programat els diferents instruments amb la mateixa veu de manera que les quatre reproduccions fossin la mateixa. S'ha comprovat que ela quatre microcontroladors reproduïen la veu a la vegada, i per tant, que estaven en correcte sincronisme.

En el CD adjunt en què també es troba la memòria del treball, hi ha una part dels vídeos mostrant la reproducció d'aquestes peces musicals.

7. Pressupost

El pressupost del projecte és degut bàsicament al cost del desenvolupament, al consum energètic durant la realització i al preu del material -majoritàriament el hardware utilitzat.

Taula 1: Pressupost corresponent al material.

Material			
Producte	Quantitat	Cost Unitari (€/unitat)	Cost Total (€)
Plaques	4	80	320
Ordinador	1	(550/5) = 110	110
ICD 2	1	(180/5) = 36	36
Altaveus	4	10	40
Hardware d'Àudio	4	0.6	2.4
Kvaser Leaf Light	1	(220/5) = 44	44
Cablejat bus CAN	5	0.2	1

Per considerar el cost dels equips utilitzats en el pressupost del projecte es calcula l'amortització d'aquests i no el seu preu total. L'amortització és el cost total dividit entre la vida útil; que s'ha considerat d'uns cinc anys.

Taula 2: Pressupost corresponent al consum energètic.

Consum Energètic			
Consum elèctric	Hores	Cost (€/kWh)	Cost Total (€)
Dispositius	360	0.14	50.4
Espai de treball	450	0.14	63

Taula 3: Pressupost corresponent al cost del desenvolupament.

Cost del Desenvolupament			
Tasca	Hores	Cost (€/hora)	Cost Total (€)
Estudi Previ	90	20	1800
Fabricació Hardware	50	20	1000
Desenvolupament Software	250	20	5000
Test i Validació	60	20	1200

Per tant, el cost total del projecte, considerant els tres grans blocs comentats explicats en la Taula 1, Taula 2 i Taula 3, s'aproxima als 9666.8 €. Com es pot observar, la major part del pressupost va destinada al desenvolupament del projecte.

8. Impacte mediambiental

L'impacte mediambiental d'un projecte és l'efecte que aquest comporta en el seu entorn a partir de la interacció amb ell. Per tant, hi ha dos aspectes a considerar en relació amb l'impacte mediambiental que causa aquest projecte. En primer lloc els residus que genera el hardware utilitzat ; i en segon lloc el consum d'energia elèctrica utilitzada durant la realització del projecte.

Els residus que pugui generar el hardware emprat en el projecte s'han de reciclar o bé portar a una planta de gestió de residus electrònics. Tenint en compte que la vida útil dels components utilitzats és relativament alta, si es fa un reciclatge adequat dels residus generats, l'impacte mediambiental es veu reduït.

L'energia elèctrica consumida durant el desenvolupament del projecte no suposa un ús excessiu de les fonts energètiques utilitzades, i per tant l'impacte mediambiental que se'n deriva no és més que l'impacte que causen les mateixes fonts.

A més, aquest projecte genera un tipus diferent d'impacte ambiental: la contaminació acústica. En l'execució del projecte es reprodueix música mitjançant uns altaveus, i per tant, existeix un nivell de contaminació acústica. Com que els altaveus utilitzats no són de volum regulable, en el disseny del hardware d'àudio s'ha procurat que el so produït pels altaveus no sigui excessivament elevat fixant el valor d'una resistència a l'entrada dels altaveus.

Conclusions

A partir de la feina realitzada en aquest treball de final de grau, s'ha aconseguit crear una orquestra electrònica amb un programa executable en un ordinador personal fent la feina de director i quatre microcontroladors PIC18F4520 fent la feina de quatre instruments; baix, violí i dues guitarres. S'han creat tres tipus d'orquestra diferents:

- Una orquestra en què s'emmagatzema la partitura d'una peca en la memòria ROM dels quatre microcontroladors i s'executa la reproducció d'aquesta peca a partir de l'execució d'un programa director que envia l'ordre mitjançant un bus CAN per sincronitzar tots els instruments.

- Una orquestra en què el programa director envia una partitura curta als quatre microcontroladors i els dóna l'ordre d'execució mitjançant un bus CAN.

- Una orquestra en què el programa director dóna ordres als instruments per reproduir la partitura d'una peça de qualsevol llargària. Primer envia una part de la partitura i a partir d'aquí es comunica amb els quatre microcontroladors a partir d'un bus CAN consultant si ha de seguir enviant més notes de la partitura o no.

- Un programa que permet aconseguir la partitura d'una peça en el format necessari per a poder ser utilitzada a partir d'un fitxer tipus midi.

El resultat obtingut servirà de base per a futurs projectes i treballs que hi puguin estar relacionats.

Agraïments

Al professor Manuel Moreno Eguilaz, director del treball de fi de grau, per haver fet aquesta proposta de treball i haver estat a disposició en tot moment.

Bibliografia

Referències bibliogràfiques

[1] TIMOFEEV, VICTOR. *OSA: Documentation.* 2007.

[http://www.pic24.ru/doku.php/en/osa/ref/intro, 10 de gener de 2016].

[2] DOMENEC, PERE. *Aplicaciones musicales del sistema operativo en tiempo real OSA RTOS.* Barcelona, 2016.

[3] CUTHBERT, MICHAEL SCOTT. CUTHBERTLAB. *music21: a toolkit for computer-aided musicology.* 2006.

[http://web.mit.edu/music21/, 10 de gener de 2016].